(Concise explanations of the disclosure)

Japanese laid-open patent publication No. 9-114734
Published on December 1, 1998

A semiconductor device may include a data cache or a data cache system and a store queue serving as a write buffer or a store buffer for data-write instruction or data store instruction. Data write operation to a main memory and data cache operation to a data memory may be made, wherein a store instruction including a write address and data is once held by the store queue for improvement in throughput of the processor.

Store buffer apparatus with two store buffers to increase throughput of a store operation

Patent Number: US5845321

Publication date: 1998-12-01

Inventor(s): UEHARA KATSUTOSHI (JP); ISOBE TOSHIKO (JP); KAMADA EIKI (JP); ITO MOTOHISA (JP); YAMAMOTO KEI (JP)

Applicant(s):: HITACHI LTD (JP); HITACHI INFORMATION TECHNOLOGY (JP)

Requested Patent: JP9114734

Application Number: US19960729837 19961015

Priority Number (s): JP19950266947 19951016

IPC Classification: G06F12/00 ; G06F13/00

EC Classification: G06F12/08B6P

Equivalents:

Abstract

A store buffer apparatus connected to a CPU and a main storage unit includes a first buffer for holding a pair of store address and store data in the main storage unit supplied from an operation execution unit of the CPU, a first latch connected to the first buffer means for holding the store address, a second latch connected to the first latch for holding an output of the first latch, a judgment device for comparing an output read out from the address array with an output of the second latch to thereby judge whether the cache hit check for the store address is successful or not and a second buffer for holding the pair of store data and store address having successful cache hit check judged by the judgment device. Occurrence of the state that the store buffer is full is reduced. Two data stored in the second buffer can possess a format into which the two data can be merged.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－114734

(43)公開日　平成９年(1997)５月２日

(51)Int.Cl.⁶	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　12/08		7623－5B	Ｇ０６Ｆ　12/08	Ｃ
	３１０	7623－5B		３１０Ｚ

審査請求　未請求　請求項の数３　ＯＬ　（全 21 頁）

(21)出願番号　特願平7－266947

(22)出願日　平成７年(1995)10月16日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(71)出願人　000233011
日立コンピュータエンジニアリング株式会社
神奈川県秦野市堀山下１番地

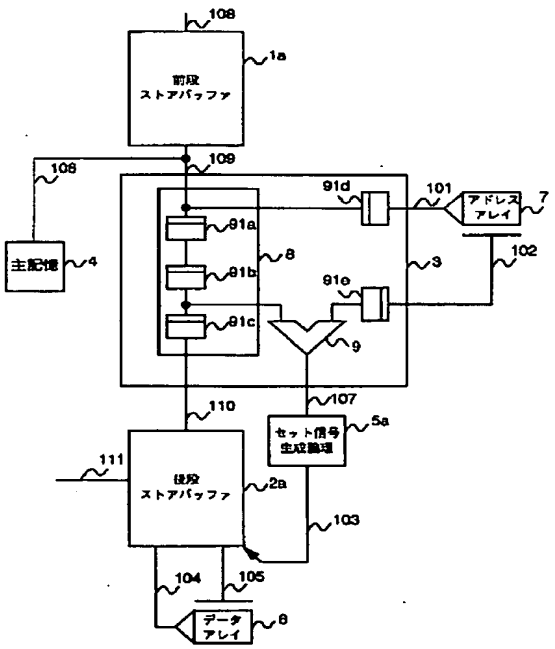
(72)発明者　伊藤　元久
神奈川県秦野市堀山下１番地　株式会社日立製作所汎用コンピュータ事業部内

(74)代理人　弁理士　鈴木　誠

最終頁に続く

(54)【発明の名称】　ストアバッファ装置

(57)【要約】
【課題】　キャッシュのヒット判定とデータ書き込みを分離することでキャッシュのヒット判定をパイプライン的に実行し、キャッシュのヒット判定の実行ピッチとストアバッファへの入力ピッチを一致させ、また、キャッシュヒットするストア命令のみストアバッファに入力することにより、ストアバッファに空きがなくなる状態の発生を低減する。
【解決手段】　ストアバッファを前段ストアバッファ１ａと、後段ストアバッファ２ａに分離し、前段ストアバッファ１ａと後段ストアバッファ２ａの間でキャッシュのヒット判定３をパイプライン的に行う。キャッシュヒットするストア命令のみを後段ストアバッファ２ａに入力する。データアレイ６への書き込みは後段ストアバッファ２ａから行う。



【特許請求の範囲】
【請求項１】　演算実行部から出力され、主記憶装置およびキャッシュメモリに書き込むストアアドレスとストアデータとの対を一時保持する複数エントリからなる第１のバッファ手段と、該第１のバッファ手段から取り出したストアアドレスとストアデータとの対が入力され、前記キャッシュメモリとのヒット判定をパイプライン的に行うヒット判定手段と、該ヒット判定の結果、キャッシュヒットと判定されたストアアドレスとストアデータとの対を一時保持する複数エントリからなる第２のバッファ手段と、を備えたことを特徴とするストアバッファ装置。
【請求項２】　前記第１、第２のバッファ手段の１エントリ全てを使用しないデータ長のストアデータを保持するために、該第１、第２のバッファ手段のエントリ内で有効なデータ位置を示す情報を記憶した手段と、前記第２のバッファ手段に入力される第１のストアデータが、該第２のバッファ手段に既に存在する第２のストアデータと同一エントリに併合可能であるか否かを判定する手段と、該手段により併合可能と判定されたとき、該第１のストアデータを該第２のストアデータと併合して該同一エントリに書き込む手段と、該書き込みに応じて、前記記憶された、第２のバッファ手段のエントリ内で有効なデータ位置を示す情報を更新する手段と、を備えたことを特徴とする請求項１記載のストアバッファ装置。
【請求項３】　前記第２のバッファ手段に入力される第１のストアデータが、該第２のバッファ手段に既に存在する第２のストアデータと連結可能であるか否かを判定する手段と、該手段により連結可能と判定されたとき、連結する前記各ストアデータのエントリ番号を記憶する手段と、該記憶されたエントリ番号を基に、前記第２のバッファ手段内の複数エントリのストアデータを連結して前記キャッシュメモリに掃き出す手段と、を備えたことを特徴とする請求項１記載のストアバッファ装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、キャッシュメモリを持ち、演算処理装置から主記憶装置及びキャッシュメモリに書き込まれるストアアドレスとストアデータを記憶するストアバッファ装置に関する。
【０００２】
【従来の技術】ストア命令が発行されると、キャッシュメモリのヒット判定を行い、キャッシュヒットしたデータがキャッシュメモリに書き込まれる。この従来の方式では、キャッシュのヒット判定の時間が必要なため、書き込み動作のピッチが長くなり、その結果、ストア命令が連続すると中央演算処理装置(以下、ＣＰＵと呼ぶ)の性能が低下する。
【０００３】この問題点を解決する方法として、データをストアバッファに一旦保持して、キャッシュメモリへの書き込み動作はストアバッファへの書き込みをもって終了し、これによりＣＰＵは、データが実際にキャッシュメモリに書き込まれるのを待たずに次の命令の処理に移ることができるキャッシュメモリ制御方法がある（特開平４－３７９３５号公報を参照）。
【０００４】上記した従来手法では、最後に書き込んだストアアドレスとキャッシュヒットの判定結果を記憶しておく。そして、後続のストアアドレスと記憶しておいたストアアドレスとを比較し、一致する場合には、キャッシュのヒット判定を行わずに、記憶しておいたキャッシュヒット判定結果を使用する。このようにして、キャッシュのヒット判定を省略することで、書き込み動作のピッチを短くして性能の向上を図っている。
【０００５】
【発明が解決しようとする課題】上記公報に記載された方法では、非連続なアドレスにストアする場合、後続のストアアドレスと記憶しておいたストアアドレスが一致せず、記憶しているキャッシュヒット判定を使用することができない。この場合には、改めてキャッシュのヒット判定をしなければならず、書き込みの動作ピッチは向上しない。そのため、ストアバッファは常に空きがない状態になり、バッファとして機能しなくなる。このような状態になると、ストアバッファに空きができるまで、ＣＰＵはその実行を待ち合わせるため、ＣＰＵの性能が低下するという問題がある。
【０００６】また、上記した従来の方法では、ストアバッファから出力される段階で、キャッシュのヒット判定が行われるので、キャッシュミスになるストア命令もストアバッファに保持されていて、ストアバッファのエントリが有効に使用されていない。
【０００７】さらに、非連続なアドレスにストアする場合、主記憶への書き込み動作ピッチも長くなり、主記憶につながるバスが有効に使用されないという問題もある。
【０００８】本発明の目的は、連続アドレスに対するストアと非連続のアドレスに対するストアの動作ピッチを同じにして、ストアバッファの入力と出力のピッチを一致させ、ストアバッファに空きがなくなる状態の発生を低減させることで、ＣＰＵの性能を低下させないようにしたストアバッファ装置を提供することにある。
【０００９】本発明の他の目的は、ストアバッファに入力する前にキャッシュのヒット判定を行い、キャッシュヒットするストア命令のみストアバッファに入力することで、ストアバッファの使用効率を向上させたストアバッファ装置を提供することにある。
【００１０】本発明のさらに他の目的は、複数のストアバッファエントリのデータを同時にキャッシュに書き込むことで、出力スループットを向上させたストアバッファ装置を提供することにある。
【００１１】本発明のさらに他の目的は、キャッシュの

ヒット判定をパイプライン的におこなうことで、主記憶への書き込みピッチとキャッシュのヒット判定のピッチを一致させ、主記憶につながるバスを有効に使用するストアバッファ装置を提供することにある。

【００１２】

【課題を解決するための手段】前記目的を達成するために、請求項１記載の発明では、演算実行部から出力され、主記憶装置およびキャッシュメモリに書き込むストアアドレスとストアデータとの対を一時保持する複数エントリからなる第１のバッファ手段と、該第１のバッファ手段から取り出したストアアドレスとストアデータとの対が入力され、前記キャッシュメモリとのヒット判定をパイプライン的に行うヒット判定手段と、該ヒット判定の結果、キャッシュヒットと判定されたストアアドレスとストアデータとの対を一時保持する複数エントリからなる第２のバッファ手段とを備えたことを特徴としている。

【００１３】請求項２記載の発明では、前記第１、第２のバッファ手段の１エントリ全てを使用しないデータ長のストアデータを保持するために、該第１、第２のバッファ手段のエントリ内で有効なデータ位置を示す情報を記憶した手段と、前記第２のバッファ手段に入力される第１のストアデータが、該第２のバッファ手段に既に存在する第２のストアデータと同一エントリに併合可能であるか否かを判定する手段と、該手段により併合可能と判定されたとき、該第１のストアデータを該第２のストアデータと併合して該同一エントリに書き込む手段と、該書き込みに応じて、前記記憶された、第２のバッファ手段のエントリ内で有効なデータ位置を示す情報を更新する手段とを備えたことを特徴としている。

【００１４】請求項３記載の発明では、前記第２のバッファ手段に入力される第１のストアデータが、該第２のバッファ手段に既に存在する第２のストアデータと連結可能であるか否かを判定する手段と、該手段により連結可能と判定されたとき、連結する前記各ストアデータのエントリ番号を記憶する手段と、該記憶されたエントリ番号を基に、前記第２のバッファ手段内の複数エントリのストアデータを連結して前記キャッシュメモリに掃き出す手段とを備えたことを特徴としている。

【００１５】上記したように、キャッシュのヒット判定とキャッシュへのデータ書き込みを分離したことにより、後段ストアバッファへの入力と、後段ストアバッファからの出力のピッチが一致し、後段ストアバッファに空きがなくなる状態の発生を低減することができる。また、キャッシュのヒット判定を後段ストアバッファ入力時に行い、キャッシュヒットするストア命令のみ後段ストアバッファに入力することで、後段ストアバッファの使用効率を向上することができる。また、後段ストアバッファの複数のエントリを同時に掃き出せるようにしたことで、後段ストアバッファに空きがなくなる状態の発生を低減することができる。さらに、キャッシュのヒット判定をパイプライン的に行うことで、主記憶にデータを書き込むピッチとキャッシュのヒット判定のピッチが一致し、主記憶につながるバスを有効に使用することができる。

【００１６】

【発明の実施の形態】以下、本発明の一実施例を図面を用いて具体的に説明する。以下の説明では、２進数を“　”で囲って表すことにする。図２５は、本発明の実施対象となる情報処理装置の構成を示す。図中、７４はＣＰＵ、７５はストアスルー方式の２次キャッシュメモリ(以下、ＳＣＭと呼ぶ)、４は主記憶装置である。７１はプログラムを実行する命令実行部、７２はＣＰＵ内蔵の１次キャッシュメモリ(以下、ＦＣＭと呼ぶ)、７３はストアバッファである。６はＳＣＭのデータアレイ、７はＳＣＭのアドレスアレイである。１０８はストアアドレスとデータ、１０６は主記憶にデータを書き込むパスである。１０１はＳＣＭアドレスアレイ７を参照するアドレス、１０２はＳＣＭアドレスアレイ７の内容、１０４はＳＣＭストアアドレス、１０５はＳＣＭストアデータである。

【００１７】情報処理装置において、メモリアクセスの高速化のためにキャッシュメモリを用いるのが一般的である。ＣＰＵチップ内に設けたＦＣＭ７２は、高速に動作するが容量を大きくするのは難しい。そこで、ＦＣＭ７２の他に、ＣＰＵチップ外にＳＣＭ７５を設けることがある。このＳＣＭ７５は、速度はＦＣＭ７２に劣るが、大容量化が可能である。図２５は、ＣＰＵ７４の外にＳＣＭ７５を設けた場合の構成例である。ＳＣＭ７５は、データアレイ６とアドレスアレイ７からなる。データアレイ６にデータを書き込み、アドレスアレイ７で内容の管理を行う。

【００１８】ストア命令の実行は、ＦＣＭヒットならばＦＣＭ７２にデータを書き込み、同時にストアバッファ７３に書き込む。ストアバッファ７３にデータが書き込まれた段階でストア命令の実行は終了し、命令実行部７１は次の命令の処理に移る。ストアバッファに保持されたストア命令は、命令実行部７１の動作とは非同期に、主記憶４とＳＣＭ７５に書き込む。ただし、ＳＣＭミスならば、ＳＣＭ７５には書き込まない。

【００１９】〈実施例１〉図１は、本発明の実施例１の構成であり、ストアバッファ、ＳＣＭ、主記憶の書き込みに関係する部分を示す。図１において、１ａは前段ストアバッファ、２ａは後段ストアバッファ、３はＳＣＭヒット判定論理、４は主記憶、５ａはセット信号生成論理である。本発明では、図２５のストアバッファ７３を、前段ストアバッファ１ａと後段ストアバッファ２ａに分割し、前段ストアバッファ１ａと後段ストアバッファ２ａとの間にＳＣＭヒット判定論理３を設けて構成したものである。

【００２０】ＳＣＭヒット判定論理３は、パイプライン遅延論理８、アドレス比較器９、ＳＣＭアドレスアレイ７の参照アドレスを保持するラッチ９１ｄ、読み出したアドレスアレイの内容を保持するラッチ９１ｅからなる。パイプライン遅延論理８は、ストアアドレスとデータをパイプライン的に遅延させるためのラッチ９１ａ～ｃで構成されている。

【００２１】１０３はセット信号生成論理５ａから出力されるセット信号、１０７はキャッシュヒット判定結果である。１０９は前段ストアバッファ１ａから出力されるストアアドレスとデータであり、１１０はパイプライン遅延論理８で遅延されたストアアドレスとデータである。１１１は、指定されたエントリを後段ストアバッファ２ａからデータアレイ６に掃きだすための掃き出し要求である。

【００２２】図１に示す実施例１の特徴は、ストアバッファが前後２段に分かれていて、前段ストアバッファ１ａと後段ストアバッファ２ａの間のＳＣＭヒット判定論理３内でパイプライン的にＳＣＭヒット判定を行い、ＳＣＭヒットしたストア命令のみを後段ストアバッファ２ａに入力することにある。

【００２３】図２は、実施例１の前段ストアバッファ１ａの構成を示す。主記憶４に書き込む順序を保障するため、前段ストアバッファ１ａは、先入れ先出し型(ＦＩＦＯ)のバッファで構成されている。図２において、前段ストアバッファ１ａは、本実施例では８段のエントリからなり、各エントリは、本実施例では３２ビットのアドレスフィールド５３、３２ビットのデータフィールド５４ａからなる。５５はリードライトポインタ、１２２はライトエントリ、１２３はリードエントリである。

【００２４】ストアアドレスとデータ１０８は、ライトエントリ１２２が示すエントリに書き込まれ、リードエントリ１２３が示すエントリのアドレスとデータをセレクタ６０ａで選択して、１０９に出力する。

【００２５】前段ストアバッファ１ａから出力したストアアドレスとデータは、主記憶４に送られ、ストアアドレスにデータが書き込まれ、それと同時にＳＣＭヒット判定論理３に入力される。ＳＣＭヒット判定論理３は、パイプライン的にＳＣＭヒット判定を行う。

【００２６】なお、本実施例のＳＣＭのアドレスマッピングは、例えばダイレクトマッピング法を用いる。図２６は、ダイレクトマッピング法を説明する図である。３２ビットのストアアドレスの内、上位の例えば１２ビット（Ａ０～Ａ１１、タグ）が比較器の一方に入力される。また、例えばストアアドレスＡ１２～Ａ２７の１６ビット（インデックス）がアドレスアレイ７の参照アドレスとなって１ＭブロックのＳＣＭが参照される。そして、下位のＡ２８～Ａ３１の４ビットがブロック内のアドレスとなる。ここでは、１ブロックが１６バイトで構成されている。

【００２７】アドレスアレイ７には上位１２ビット（タグ）が格納され、参照アドレスで読み出される上位１２ビットが比較器の他方の入力となる。ストアアドレス中のインデックスを用いてアドレスアレイ７のエントリを指定し、該エントリのタグを読み出す。ストアアドレス中のタグとアドレスアレイ７のタグが比較され（このとき、タグに付けられている有効ビットもチェックされる）、一致したとき、１ブロックまたは、ストアアドレス中のブロック内アドレスで指定された１バイトが読み出される。また、タグが一致しないときは、主記憶をアクセスする。

【００２８】図３は、ＳＣＭヒット判定論理３の動作例を示す。ラッチ９１ａ～ｃで構成されたパイプライン遅延論理８で、ストアアドレスとデータをパイプライン的に遅延する。また、ラッチ９１ｄは、アドレスアレイ７の参照アドレス（つまり、上記したインデックス）を保持する。ＳＣＭの一実施例として、図３に示すように、アドレスアレイ参照アドレスを与えてから、３サイクル後にアドレスアレイ７の内容（つまり、上記したタグ）が読み出されるＳＣＭを考える。このＳＣＭは、２サイクルピッチでアドレスを与えることで連続してデータの読み出しが可能なラッチ付きスタチックラム(Ｌａｔｃｈｅｄ　ＳＲＡＭ)で構成されているものとする。アドレスアレイ７の内容がラッチ９１ｅに読み出されるまで、ストアアドレスはラッチ９１ａ～ｂで遅延する。ラッチ９１ｂのストアアドレス（タグ）とラッチ９１ｅのアドレスアレイ７の内容（タグ）をアドレス比較器９で比較し、ＳＣＭヒット判定を行う。ＳＣＭヒット判定結果１０７は、

ＳＣＭヒット：“１”
ＳＣＭミス　：“０”

の値になる。ＳＣＭヒット判定結果１０７＝“１”（ＳＣＭヒット)ならば、セット信号生成論理５ａで後段ストアバッファ２ａのセット信号１０３を生成する。

【００２９】このように、本発明ではＳＣＭヒット判定をパイプライン的に行っているので、アドレスアレイ７からアドレスアレイの内容１０２が出力されると、直ちに前段ストアバッファ１ａから次のストア命令をＳＣＭヒット判定論理３に投入することができ、この結果、従来のものに比べてキャッシュのビジー率が低下する。

【００３０】図４は、セット信号１０３を生成するセット信号生成論理５ａの構成を示す図である。５１は後段ストアバッファ２ａのライトポインタ用ラッチ、５２は更新論理である。ラッチ５１は０から７の値を取り、更新論理５２はラッチ５１の値を＋１する。なお、ラッチ５１の値が７のとき、更新論理５２は０を出力する。６９はデコーダで、ラッチ５１の値により以下の値を出力する。

【００３１】

ラッチ５１の値	デコーダ６９の出力
0	“１０００００００”
1	“０１０００００”
2	“００１００００”
3	“０００１０００”
4	“００００１０００”
5	“０００００１００”
6	“００００００１０”
7	“０００００００１”

１０３は、後段ストアバッファ２ａに入力するセット信号である。セット信号１０３は、本実施例では後段ストアバッファ２ａが８段エントリであるので８ビットで構成され，“１”が立っている後段ストアバッファ２ａのエントリにストアアドレスとストアデータ、および有効ビットが入力される。つまり、セット信号１０３の０ビット目は０番エントリに入力することを示し、以下順に、７ビット目が７番エントリに入力することを示す。例えば、３番エントリに入力するセット信号１０３は、“０００１０００”になる。セット信号１０３がオール“０”のときは、後段ストアバッファ２ａのどのエントリにも入力されない。このセット信号１０３は、以下のように生成される。

【００３２】（１）ＳＣＭヒット判定結果１０７＝“１”（ＳＣＭヒット）のとき、セレクタは更新論理５２を選択し、従ってセット信号１０３としてデコーダ６９の出力がアンドゲートを介して出力される。また、ラッチ５１の値は更新論理５２により更新される。

【００３３】（２）ＳＣＭヒット判定結果１０７＝“０”（ＳＣＭミス）のとき、セレクタは“０”を選択し、従ってセット信号１０３としてオール“０”が出力され、ラッチ５１の値は更新されない。

【００３４】図５は、前段ストアバッファ１ａ内のリードライトポインタ５５の構成を示す図である。５６ａはライトポインタ用ラッチ、５６ｂはリードポインタ用ラッチ、５２は前述した更新論理、６９は前述したデコーダである。１２８は命令実行部からのライト要求信号、１２９は命令実行部からのリード要求信号である。

【００３５】本実施例では前段ストアバッファ１ａが８段のエントリからなるので、ライトポインタ用ラッチ５６ａ、リードポインタ用ラッチ５６ｂともに０から７の値をとる。ライト要求信号１２８が“１”になるたびに、前段ストアバッファ１ａにストアアドレス、データが書き込まれ、ライトポインタ用ラッチ５６ａは次の値を示すように更新される。また、リード要求信号１２９が“１”になるたびに、前段ストアバッファ１ａからストアアドレス、データが掃き出され、リードポインタ用ラッチ５６ｂは次の値を示すように更新される。なお、更新論理５２は、ライトポインタ用ラッチ５６ａ、あるいは、リードポインタ用ラッチ５６ｂの値が７のときは０を出力する。

【００３６】図６は、実施例１の後段ストアバッファ２ａの構成を示す図である。図６において、本実施例では後段ストアバッファ２ａは、８段のエントリからなり、各エントリは、本実施例では３０ビットのアドレスフィールド５７ａ、３２ビットのデータフィールド５８a、１ビットの有効ビット５９からなる。

【００３７】アドレスフィールドが前段ストアバッファ１ａと同じ３２ビット構成でないのは、本実施例ではデータフィールド５８ａが３２ビット構成（４バイト）であり、かつ、アライン（アドレス調整）されているので、ストアアドレスの下位２ビット（３０、３１ビット目）はＳＣＭ書き込み時に使用しないためである。つまり、データアレイの１ブロックが４バイト構成されている。

【００３８】６１は後段ストアバッファ２ａのリードポインタ、６２はリードポインタ用ラッチ、５２は前述した更新論理、６９は前述したデコーダである。ラッチ６２は０から７の値をとり、更新論理５２はラッチ６２の値を＋１する。なお、ラッチ６２の値が７のとき、更新論理５２は０を出力する。

【００３９】１０３は前述したセット信号生成論理５ａからのセット信号、１１１は命令実行部からの掃き出し要求信号、１０４はＳＣＭデータアレイの書き込みアドレス、１０５はＳＣＭデータアレイの書き込みデータである。

【００４０】パイプライン遅延論理３で遅延したストアアドレスとデータは、信号線１１０から入力され、ストアアドレスはアドレスフィールド５７ａ内のセット信号１０３で示すエントリに入力され、データはデータフィールド５８ａ内のセット信号１０３で示すエントリに入力される。同時に、当該エントリの有効ビット５９を“１”にセットする。セット信号１０３がオール“０”ならば、どのエントリにもストアアドレスとデータは入力しない。このとき、有効ビット５９は変化しない。すなわち、ＳＣＭミスになるストア命令は、後段ストアバッファ２ａには入力されない。

【００４１】一方、後段ストアバッファ２ａからＳＣＭへの掃き出しは、掃き出しエントリ３０６の示すエントリから行われる。掃き出しエントリ３０６は、本実施例では前段ストアバッファ２ａが８段のエントリから構成されているので、８ビットで構成される。掃き出しエントリ３０６の１ビット目は１番エントリを掃き出すことを示し、以下順に、７ビット目が７番エントリから掃き出すことを示す。例えば、０番エントリを掃き出す掃き出しエントリ３０６の値は“１００００００”になる。掃き出しエントリ３０６は、以下のように生成される。

【００４２】（１）デコーダ６９の出力＝“１”かつ、
有効ビット５９＝“１”（有効）、

かつ、
掃き出し要求１１１＝“１”（有効）、
ならば、
掃き出しエントリ３０６の該当ビット＝“１”
（２）上記以外
掃き出しエントリ３０６の該当ビット＝“０”
セレクタ６０ｃは、掃き出しエントリ３０６が示すエントリのアドレスフィールド５７ａからストアアドレスを選択して信号線１０４に出力し、データフィールド５８ａからデータを選択して信号線１０５に出力する。掃き出しエントリ３０６がオール“０”ならばどのエントリも出力しない。ストアアドレスとデータの掃き出しと同時に、掃き出したエントリの有効ビット５９をリセットする。さらに、ラッチ６２の値は更新論理５２により更新する。

【００４３】掃き出し要求１１１は、後段ストアバッファａにストア命令があり、かつ、ＳＣＭが動作していないときに、ＳＣＭコントローラ（図１には記載されていない）から出力する。このＳＣＭコントローラは公知技術で実現可能である。

【００４４】図７は、図１の実施例１において、連続してストア命令を処理するときのタイムチャートを示す。本実施例１では、ストア命令は、Ｓｔｏｒｅ－１からＳｔｏｒｅ－３まで３命令を連続して前段ストアバッファ１ａから掃き出す場合を例にしている。そして、Ｓｔｏｒｅ－１、３はＳＣＭヒットし、Ｓｔｏｒｅ－２はＳＣＭミスしたものとする。また、本実施例では信号線１０６は、２サイクルピッチで主記憶へのストア命令を受け付け可能であるものとする。

【００４５】以下、実施例１の動作を説明すると、主記憶に書き込むのと同じピッチで、ラッチ９１ａにストアアドレスとデータが入力され、ラッチ９１ｄにアドレスアレイの参照アドレス（ストアアドレス中のインデックス）が入力される。インデックスで指定されたアドレスアレイ７の内容（つまり、タグ）は、３サイクル後にラッチ９１ｅに読み出される。すなわち、ラッチ９１ｅの値は２サイクルピッチで変化する。

【００４６】ラッチ９１ｅのアドレスアレイの内容（タグ）とラッチ９１ｃの遅延したストアアドレス中のタグをアドレス比較器９で比較して、ＳＣＭヒット判定結果１０７が生成される。その結果、セット信号１０３、ラッチ５５の値は以下のように変化する。

【００４７】

Ｓｔｏｒｅ－１　ＳＣＭ　ヒット：
　セット信号１０３＝“１０００００００”：ラッチ５５＝１
Ｓｔｏｒｅ－２　ＳＣＭ　ミス：
　セット信号１０３＝“００００００００”：ラッチ５５＝１
Ｓｔｏｒｅ－３　ＳＣＭ　ヒット：
　セット信号１０３＝“０１０００００”：ラッチ５５＝２

従って、Ｓｔｏｒｅ－１のストアアドレスとデータは、セット信号１０３が指示する後段ストアバッファ２ａ内の、０番目のエントリのアドレスフィールド５７ａ、データフィールド５８ａにそれぞれ保持され（また、同時に有効ビット５９もセットされる）、同様に、Ｓｔｏｒｅ－３のストアアドレスとデータは、後段ストアバッファ２ａ内の１番目のエントリに保持され、ラッチ５５の値は２になる。

【００４８】後段ストアバッファ２ａに保持されているストア命令は、命令実行部からの掃き出し要求１１１によって、後段ストアバッファ２ａからＳＣＭのデータアレイ６に書き込まれる。すなわち、リードポインタ６１によって指定されたエントリがセレクタ６０ｃで選択される。選択されたエントリのアドレスフィールド５７ａがデータアレイ書き込みアドレスとなってデータアレイ６をアクセスする。ここで、データアレイ書き込みアドレスは、下位２ビットを除く上位３０ビットであり、該アドレスで指定されたブロック内に、先のリードポインタ６１で指定されたエントリのデータ（３２ビット）が書き込まれる。なお、図１のＳＣＭは、図２６のキャッシュメモリと異なり、アドレスアレイ７とデータアレイ６に分離しているので、データアレイ６にはアドレスが割り付けられている。

【００４９】〈実施例２〉次に、実施例２について説明する。通常、整数型のデータは４バイト（３２ビット）幅であることが多いが、浮動小数点データは、倍精度であると８バイト（６４ビット）の幅が必要である。浮動小数点データを１エントリに格納できるように、データフィールドの幅を８バイトにすると、整数型のデータを格納する際に空きができて無駄が生じる。

【００５０】そこで、データフィールドの幅を８バイトにし、さらに、同じエントリを使用するストア命令は、併合して１つのエントリを使用するようにすれば、浮動小数点データを１エントリに格納できるうえに、整数型のデータを格納する際の無駄が生じない。同じエントリを使用するストア命令を併合して１つのエントリを使用することを、本実施例ではデータの併合と呼ぶことにする。そして、データの併合は、後段ストアバッファに対して行う。つまり、既に後段ストアバッファに存在するストアデータと、後段ストアバッファに入力されるストアデータが同一エントリに併合可能であるとき、データの併合を行う。

【００５１】データが併合可能か否かは、データフィールドが８バイトであるから、下位３ビットを除く、スト

アアドレスの上位２９ビット（０～２８ビット）を比較することによって同一のエントリであるか否かが判明する。ストアアドレスの比較の結果、データ併合が可能ならば、既存のデータフィールドに新しいストアデータを書き込む。ただし、主記憶４に書き込む順番を保障する目的を持つ前段ストアバッファ１ｂは、データフィールドの幅を８バイトにしてもデータの併合を行わない。

【００５２】データフィールドの幅を８バイトに拡張したので、データフィールド内のどこに有効なデータが存在するかを表す情報を記憶しておく必要が生じる。そこで、有効なデータが存在するバイト位置を表す情報をデータフィールドの各エントリ毎に設ける。これをバイトマスクと呼ぶ。

【００５３】バイトマスクの値は，

“１”：有効なデータが存在する

“０”：有効なデータが存在しない

である。例えば、０バイト目から３バイト目まで有効なデータが存在する場合のバイトマスクは、“１１１１００００”となる。

【００５４】図８は、本発明の実施例２の構成を示す。実施例１の構成に、前段バイトマスクバッファ１１、後段バイトマスクバッファ１２、バイトマスク遅延論理１３、バイトマスク更新論理１４、アドレス一致検出論理１５ａを付加して構成されている。

【００５５】図９は、実施例２の前段ストアバッファ１ｂの構成を示す図である。主記憶４に書き込む順序を保障するため、前段ストアバッファ１ｂは先入れ先出し型（ＦＩＦＯ）のバッファで構成され、本実施例では８段のエントリを持つ。そして、データフィールド５４ｂは、本実施例では８バイト（６４ビット）幅である。信号線１２２と信号線１２３は，前段バイトマスクバッファ１１の書き込み、あるいは掃き出しに使用する。また、信号線１２８、１２９は、命令実行部からのライト要求、リード要求である。リードライトポインタ５５、セレクタ６０ｂは、実施例１で説明したものと同様の機能を持つ。

【００５６】図１０は、前段バイトマスクバッファ１１の構成を示す図である。データフィールド５４ｂは、本実施例では１エントリ８バイト構成であるので、バイトマスクの１エントリは８ビットで構成される。そして、前段ストアバッファ１ｂと同じ８段のエントリを持つ。信号線１１２から入力されたバイトマスクは、ライトエントリ１２２の示すエントリに入力される。掃き出しは、前段ストアバッファ１ｂの掃き出しと同時に、セレクタ６０ｆでリードエントリ１２３の示すエントリのバイトマスクを選択する。

【００５７】前段バイトマスクバッファ１１から掃き出したバイトマスク１１３は、バイトマスク遅延論理１３でＳＣＭヒット判定が終わるまで遅延させ、バイトマスク更新論理１４を通過した後、後段バイトマスクバッファ１２に入力される。

【００５８】図１１は、バイトマスク更新論理１４の構成を示す図である。１１４はバイトマスク遅延論理１３で遅延させたバイトマスク、１１６－０～７は、後段バイトマスクバッファ１２に保持されたバイトマスクで、１１６－０は０番エントリのバイトマスク、以下順に、１１６－７が７番エントリのバイトマスクである。１１５－０～１１５－７は後段バイトマスク１２の入力である。１１５－０は０番エントリの入力であり、以下順に、１１５－７が７番エントリの入力である。１２１はデータ併合の有無を示す信号で、

“０”：併合なし、

“１”：併合あり、

を表す。

【００５９】データ併合の有無に応じて、後段バイトマスクバッファ１２の入力１１５－０～７を以下のように生成する。すなわち、

（１）信号線１２１＝“０”（データ併合なし）

後段バイトマスク１２の入力＝ストアデータのバイトマスク１１４

（２）信号線１２１＝“１”（データ併合あり）

後段バイトマスク１２の入力＝ストアデータのバイトマスク１１４　ｏｒ（論理和）既存のバイトマスク１１６－０～７

図１２は、後段バイトマスクバッファ１２の構成を示す図である。後段バイトマスクバッファ１２は、本実施例では、８エントリで構成され、各エントリは８ビットで，データフィールド５８ｂ内の有効なデータが存在するバイト位置を示す。１１５－０～７は各エントリの入力であり、１１６－０～７は各エントリの出力である。前段バイトマスクバッファ１１と違い、入力は共通ではなく各エントリ毎に個別になっている。そして、セット信号１０３の示すエントリのバイトマスクのみが書き込まれ、その他のエントリのバイトマスクは変化しない。

【００６０】図１３は、実施例２の後段ストアバッファ２ｂの構成を示す図である。各エントリは本実施例２では、２９ビットのアドレスフィールド５７ｂ、６４ビット（８バイト）のデータフィールド５８ｂ、１ビットの有効ビット５９からなる。アドレスフィールドが３２ビット構成でないのは、データフィールド５８ｂが６４ビット構成であるので、つまり、ＳＣＭデータアレイの１ブロックが６４ビット（８バイト）であり、ブロックエントリがストアアドレスの上位２９ビットで指定され、下位３ビット（２９～３１ビット目）がＳＣＭデータアレイの書き込み時に使用されないからである。

【００６１】６０ｄは、アドレスフィールド５７ｂ、データフィールド５８ｂから掃き出すエントリを選択するセレクタ、６３ａは、有効なデータが存在するバイト位置のストアデータを取り出すバイトセレクタである。

【００６２】１０３はセット信号、１０４はデータアレ

イ書き込みアドレス、１０５は書き込みデータである。１１０はパイプライン遅延論理３で遅延したストアアドレスとデータ、１１４はバイトマスク遅延論理１４の出力、１１１は掃き出し要求である。１１６－０～７は後段バイトマスクバッファ１２の内容である。１１９－０～９はアドレスフィールド５７ｂの内容である。１２４－０～７は有効ビット５９の値である。

【００６３】データフィールド５８ｂは、バイト毎にストアデータをセット可能なように構成され、データフィールド５８ｂのセット信号３０７－０～７は、セット信号３０７－０が０バイト目のセット信号であり、以下順に、セット信号３０７－７が７バイト目のセット信号である。

【００６４】パイプライン遅延論理３で遅延されたストアアドレスとデータは、信号線１１０から入力され、ストアアドレスはセット信号１０３で示すアドレスフィールド５７ｂのエントリに入力され、データはセット信号３０７－０～７で示すデータフィールド５８ｂのエントリのバイト位置に入力される。ストアアドレスとデータが入力されると同時に、当該エントリの有効ビット５９が“１”にセットされる。セット信号１０３がオール“０”ならば、どのエントリにもストアアドレスとデータは保持されず破棄される。このとき、有効ビット５９は変化しない。すなわち、ＳＣＭミスになるストア命令は、後段ストアバッファ２ｂに入力されない。

【００６５】一方、後段ストアバッファ２ｂからの掃き出しは、掃き出しエントリ３０６が示すエントリからおこなわれる。セレクタ６０ｄは掃き出しエントリ３０６を基に、アドレスフィールド５７ｂからストアアドレスを選択して信号線１０４に出力し、データフィールド５８ｂからデータを選択して信号線３０１に出力する。ストアアドレスとデータの掃き出しと同時に、掃き出したエントリの有効ビット５９をリセットする。さらに、ラッチ６２の値は更新論理５２により更新する。バイトセレクタ６３ａは、有効なストアデータを取り出し信号線１０５に出力する。

【００６６】掃き出し要求信号１１１は、後段ストアバッファ２ｂにストア命令があり、かつ、ＳＣＭが動作していないときに、前述の公知技術であるＳＣＭコントローラ（図８には示されていない）から出力する。

【００６７】図１４は、バイトセレクタ６３ａの構成を示す図である。６０ｇは掃き出しエントリ３０６を基に、後段バイトマスクバッファの内容１１６－０～７から掃き出すエントリのバイトマスク３０８を取り出すセレクタである。このバイトマスク３０８を基にセレクタ６０ｈは、ストアデータ３０１から有効なデータを取り出す。

【００６８】図１５は、データの併合を検査するためのアドレス一致検出論理１５ａの構成を示す図である。６４－０～７は、２９ビットのアドレス比較器である。１１７は後段ストアバッファ２ｂに入力可能なストアデータのストアアドレス、１１９－０～７は、後段ストアバッファ２ｂに既に存在するストアデータのアドレスフィールド５７ｂの内容、１２４－０～７は有効ビット５９の内容である。

【００６９】２９ビットのストアアドレス１１７が、各アドレス比較器６４－０～７の一方に入力され、各アドレス比較器６４－０～７の他方にはそれぞれアドレスフィールド５７ｂの内容Ａ０～Ａ７が１１９－０～７として入力され、ストアアドレス１１７を持つストアデータが、後段ストアバッファ２ｂに既に存在するどのストアデータと併合が可能であるか否かを調べる。１１８－０～７は、アドレスが一致しデータ併合可能なことを示す。つまり、１１８－０は、ストアアドレス１１７を持つストアデータが、後段ストアバッファ２ｂの０番エントリのデータと併合可能であることを示し、以下順に、１１８－７は７番エントリとデータ併合可能であることを示す。

【００７０】ただし、アドレス比較器６４－０～７は、ｅｎａｂｌｅ（有効ビット５９）が“１”（有効）であるとき、ストアアドレス１１７の０～２８ビットと、アドレスフィールド５７ｂの内容１１９－０～７をそれぞれ比較する。そして、アドレスが一致したならば、“１”を出力し、不一致ならば“０”を出力する。ｅｎａｂｌｅ（有効ビット５９）が“０”（無効）であるときは“０”を出力する。

【００７１】図１６は、実施例２のセット信号生成論理５ｂの構成を示す図である。１２１は、図１５のデータ併合可能なことを示す信号１１８－０～７をＯＲゲート２０１を介した信号であり、“１”のときデータ併合があり、“０”のときデータ併合がないことを示す。

【００７２】セット信号生成論理５ｂは、条件により以下のように動作する。すなわち、

（１）信号線１０７＝“１”（ＳＣＭヒット）、かつ、信号線１２１＝“０”（データ併合なし）ならば、

アンドゲート２０２の出力“１”によって、セレクタ２０３は更新論理５２側の出力を選択する。ラッチ５１の値がデコーダ６９で出力され、アンドゲート２０４、オアゲート２０５を介して信号１０３に出力される。この場合はデータの併合がないので、実施例１と同様に、信号１０３で指示された後段ストアバッファ２ｂのエントリに入力される。また、この入力時に、前段バイトマスクバッファ１１からのバイトマスク１１４が後段ストアバッファ２ｂにアンドゲート４０１を介して入力されるので、そのバイトマスク１１４で指示されたバイト位置に入力される。また、ラッチ５１の値は更新論理５２により更新される。

【００７３】

（２）信号線１０７＝“１”（ＳＣＭヒット）、かつ、信号線１２１＝“１”（データ併合あり）ならば、

アンドゲート２０２の出力は“０”となり、セレクタ２０３は“０”を選択し、デコーダ６９はオール“０”となる。また、アンドゲート２０４の出力が“０”となる。従って、信号１０３としては、エントリ番号を示す信号１１８－０～７がオアゲート２０５を介して出力される。

【００７４】エントリ番号を示す信号１０３は、図１３の後段ストアバッファ２ｂに入力され、これと同時にストアデータのバイトマスク１１４が入力される。これにより、併合されるストアデータは、信号１０３で指示されたエントリ中の、バイトマスク１１４で指示されたバイト位置のデータフィールド５８ｂに入力される。

【００７５】なお、併合時に、入力ストアデータが、後段ストアバッファ２ｂに存在する併合相手のストアデータと同じバイト位置に書き込まれ、つまり上書きされることもあるが、同一エントリに対するストア命令が連続したとき、最後のストア命令が後段ストアバッファ２ｂに保持され、その後、ＳＣＭに最新のデータが掃きだされればよいので、何ら問題ではない。

【００７６】

（３）信号線１０７＝“０”（ＳＣＭミス）ならば、アンドゲート２０２の出力は“０”となり、セレクタ２０３は“０”を選択し、デコーダ６９はオール“０”となる。従って、信号線１０３はオール“０”、ラッチ５１の値は更新されない。

【００７７】〈実施例３〉次に実施例３について説明する。ＳＣＭへの書き込みデータ幅を広くすれば、データ書き込みの効率はよくなる。そこで、本実施例ではＳＣＭへの書き込みデータ幅を２倍の１２８ビット（１６バイト）に広げ、後段ストアバッファの２エントリを連結して掃き出すようにする。連結可能であるか否かは（つまり、２エントリのアドレスが連続しているか否かは）、書き込みデータ幅が１６バイトであるから、ストアアドレスの上位２８ビット（０～２７ビット目）を比較すれば判明する。

【００７８】図１７は、本発明の実施例３の構成を示した図である。２ｃは後段ストアバッファであり、１５ｂはアドレス一致検出論理である。１６は連結エントリバッファである。連結エントリバッファ１６は、後段ストアバッファ２ｃの複数のエントリを連結して掃き出すとき、どのエントリと連結するかを示す。

【００７９】実施例３の特徴は、２エントリを連結して同時に後段ストアバッファ２ｃから掃き出す制御方法にある。まず、どのエントリと連結可能であるかを示す連結エントリバッファ１６を持つ。また、ストアアドレスを比較して連結可能なエントリを判定するアドレス一致検出論理１５ｂを持つ。さらに、任意の２エントリを同時に掃き出し可能な後段ストアバッファ２ｃを持つ。

【００８０】図１８は、連結エントリバッファ１６の構成を示す図である。連結エントリバッファ１６は、後段ストアバッファ２ｃと同じ数のエントリを持ち、本実施例では８段のエントリから構成され、各エントリは８ビットの連結エントリフィールドＭＥｎからなる。

【００８１】連結エントリフィールドＭＥｎは、ｎ番エントリと連結可能なエントリを示し、０ビット目が０番エントリと連結可能であることを示し、以下順に、７ビット目が７番エントリと連結可能であることを示す。

【００８２】例えば、１番エントリと３番エントリが連結可能ならば、連結エントリフィールドＭＥ１には、３番エントリと連結可能であることを示す、　ＭＥ１＝“００００１０００”が格納され、連結エントリフィールドＭＥ３には、１番エントリと連結可能であることを示す、　ＭＥ３＝“０１０００００”が格納される。また、どのエントリとも連結できないときは、ＭＥｎ＝オール“０”になる。１２５－０～７は、アドレス一致検出論理１５ｂの出力であり、連結可能な既存のエントリを示し、１２５－０は０番エントリと連結可能を示し、以下順に、１２５－７が７番エントリと連結可能を示す。

【００８３】図１８中の表は、入力Ａ（信号１０３）、Ｂ（信号１２５－０～７）による出力Ｏの値を示す。例えば、ＡＢ＝“００”のとき、Ｏ＝０になり、ＡＢ＝“０１”のとき、Ｏ＝１になる。Ｏ＝０、１、２に応じて、セレクタ５０１は入力“０”、入力“１”（１０３）、入力“２”（１２５－０～７）を選択する。

【００８４】１０３（０）は、信号線１０３の０ビット目を表し、以下順に、（７）は７ビット目を表す。信号線１０３は、セット信号生成論理５ｂ（図１７）から出力されるもので、前述した実施例２と同様に（図１６）、後段ストアバッファ２ｃに入力されるストアデータのエントリの番号が出力される。例えば、信号線１０３の０ビット目が“１”であるときは、入力ストアデータは、データフィールド５８ｂの０番エントリに入力される。

【００８５】ＭＥｎの値は、上記したセット信号１０３、信号線１２５－０～７の値により以下のようにセットされる。なお、！＝は等しくないことを表す。

（１）信号線１０３＝オール“０”（つまり、後段ストアバッファに入力がない）

このとき、信号線１２５－０～７＝オール“０”、すなわち、ＡＢ＝“００”によって、Ｏ＝“０”となり、ＭＥｎは過去の値を保持する。

（２）信号線１０３！＝オール“０”かつ、信号線１２５－０～７＝オール“０”（つまり、後段ストアバッファに入力があり、連結エントリがない場合）

（ａ）信号線１０３＝“１”のエントリは、ＡＢ＝“１０”であるから、Ｏ＝２となり、セレクタ５０１は、信号線１２５－０～７の値を選択し、従って、ＭＥｎ＝オール“０”となる。

（ｂ）信号線１０３＝“０”のエントリは、ＡＢ＝“０

０”であるから、Ｏ＝０となり、セレクタ５０１は、“０”を選択し、ＭＥｎは過去の値を保持する。
（３）信号線１０３!＝オール“０”、かつ、信号線１２５－０～７！＝オール“０”（つまり、後段ストアバッファに入力があり、連結エントリがある場合）
（ａ）信号線１０３＝“１”のエントリは、ＡＢ＝“１０”であるから、Ｏ＝２となり、セレクタ５０１は、信号線１２５－０～７の値を選択し、従って、ＭＥｎは連結エントリを保持する。例えば、信号線１０３(０)＝“１”、つまり０番エントリが“１”であるとき、ＭＥ０が選択されて、例えば信号線１２５－１が“１”（つまり、連結可能なエントリが１番エントリ）であれば、ＭＥ０には、ＭＥ０＝“０１０００００”がセットされる。
（ｂ）信号線１０３＝“０”かつ、信号線１２５－０～７＝“０”のエントリは、ＡＢ＝“００”であるから、Ｏ＝０となり、セレクタ５０１は、“０”を選択し、ＭＥｎは過去の値を保持する。
（ｃ）信号線１０３＝“０”かつ、信号線１２５－０～７！＝“０”のエントリは、ＡＢ＝“０１”であるから、Ｏ＝１となり、セレクタ５０１は、セット信号１０３－０～７を選択し、ＭＥｎにセット信号１０３－０～７を保持する。上記した（３）の（ａ）の例では、信号線１２５－１＝“１”であるので、１番エントリのＭＥ１が選択され、ＭＥ１にセット信号１０３（０）、つまり０番エントリがセットされ、ＭＥ１＝“１００００００”となる。
【００８６】この結果、上記した例では、連結エントリバッファ１６には、
ＭＥ０＝“０１０００００”
ＭＥ１＝“１００００００”
がセットされ、０番エントリと１番エントリが連結可能であることが示される。
【００８７】図１９は、実施例３のアドレス一致検出論理１５ｂの構成を示す図である。データ併合が可能であるか否かはストアアドレスの上位２９ビットを比較すればよい。また、データ書き込み幅が本実施例では１２８ビット（１６バイト）であるので、連結可能であるか否かはストアアドレスの上位２８ビットを比較すればよい。
【００８８】６４－０～７は、データ併合が可能であるか否かを判定する２９ビットのアドレス比較器、６５－０～７は、連結可能であるか否かを判定する２８ビットのアドレス比較器である。１１７はストアアドレス、１１９－０～７は、アドレスフィールド５７ｂの内容、１２４－０～７は有効ビット５９の内容である。１１８－０～７は、アドレスが一致しデータ併合が可能なエントリを示す。１２５－０～７は、アドレスが一致し連結可能なエントリを示す。
【００８９】アドレス比較器６４－０～７は、ｅｎａｂｌｅ（１２４－０～７）が“１”（有効）であるとき、ストアアドレス１１７の０～２８ビットと、アドレスフィールド５７ｂの内容１１９－０～７をそれぞれ比較する。アドレスが一致したならば、“１”を出力し、不一致ならば“０”を出力する。ｅｎａｂｌｅが“０”（無効）の時は“０”を出力する。
【００９０】同様に、アドレス比較器６５－０～７は、ｅｎａｂｌｅ（１２４－０～７）が“１”（有効）であるとき、ストアアドレス１１７の０～２７ビットと、アドレスフィールド５７ｂの内容１１９-０～７の０～２７ビットをそれぞれ比較する。アドレスが一致したならば、“１”、不一致ならば“１”を出力する。ｅｎａｂｌｅが“０”（無効）の時は“０”を出力する。
【００９１】図２０は、実施例３の後段ストアバッファ２ｃの構成を示す図である。実施例２の構成にさらに、データセレクタ６８と、バイトセレクタ６３ｂを付加して構成されている。データセレクタ６８への入力１０４（２８）は、信号線１０４（ＳＣＭストアアドレス）の２８ビット目を表す。
【００９２】図２１は、データセレクタ６８の構成を示す図である。６０ｋ、６０１は、データフィールド５８ｂの出力３０５を選択するセレクタであり、６０ｍは連結エントリのセレクタであり、６７はリードポインタ入れ換え論理である。１０４（２８）はストアアドレス１０４の２８ビット目の値（“１”または“０”）である。３０２－０～１は選択されたストアデータ、３０３－０～１は、データセレクタ６８内のリードポインタであり、３０３－０が下位側のリードポインタ、３０３－１が上位側のリードポインタである。
【００９３】同時に２つのエントリを掃き出し可能なように、データセレクタは２個設けられている。すなわち、データセレクタ６０ｋは下位アドレスのデータを選択し、データセレクタ６０１は上位アドレスのデータを選択する。
【００９４】同時に掃き出すエントリは、掃き出しエントリ３０６が示すエントリ（これは掃き出し要求１１１によって決まる）と、そのエントリ３０６と連結可能なエントリである。セレクタ６０ｍは、掃き出しエントリ３０６の値を基に信号線１２６－０～７を選択して、当該エントリと連結可能なエントリ番号を出力する。つまり、掃き出しエントリ３０６の値を持つ連結エントリフィールドＭＥｎを参照して、掃き出しエントリ３０６と連結可能なエントリ番号を出力する。
【００９５】本実施例では、ＳＣＭのデータ書き込みパスは１２８ビット（１６バイト）幅であり、後段ストアバッファ２ｃのデータフィールドが６４ビット（８バイト）幅であるので、ストアアドレスによってデータをＳＣＭのデータ書き込みパスの上位側アドレスに載せる場合と、下位側アドレスに載せる場合がある。
【００９６】掃き出しエントリ３０６が示すエントリの

ストアアドレスは、図２０に示すセレクタ６０ｊによって選択され、ストアアドレス１０４に出力される。このストアアドレス１０４の２８ビット目の信号線１０４（２８）の値が“１”のとき、掃き出しエントリ３０６のデータを上位側に載せる。そして、連結エントリ（信号線１２６－０～７）が示すエントリのデータは下位側に載せる。

【００９７】逆に、信号線１０４（２８）の値が“０”ならば、掃き出しエントリ３０６が示すエントリのデータを下位側に、連結エントリが示すエントリのデータを上位側に載せる。

【００９８】リードポインタ入れ換え論理６７は、上記の条件に基づいてデータセレクタ６８内のリードポインタ３０３－０～１を生成する。連結エントリは、前述したように、セレクタ６０ｍによって掃き出しエントリ３０６を基に、連結エントリバッファ１６の出力１２６－０～７を選択する。リードポインタ入れ換え論理６７の動作は以下の通りである。

【００９９】（１）信号線１０４（２８）＝“１”のとき（入力３０６、１２６－０～７の“１”側の入力を選択）

リードポインタ３０３－０には、セレクタ６０ｍの出力（連結エントリ１２６－０～７）が選択され、セレクタ６０ｋに出力され、リードポインタ３０３－１には、掃き出しエントリ３０６が選択されてセレクタ６０ｌに出力される。従って、掃き出しエントリ３０６のデータ５８ｂがセレクタ６０ｌで選択されて上位側に置かれ、連結エントリのデータ５８ｂがセレクタ６０ｋで選択されて下位側に置かれ、データ３０２－０～１となる。

【０１００】（２）信号線１０４（２８）＝“０”のとき（入力３０６、１２６－０～７の“０”側の入力を選択）

リードポインタ３０３－０には、掃き出しエントリ３０６が選択されてセレクタ６０ｋに出力され、リードポインタ３０３－１には、セレクタ６０ｍの出力(連結エントリ)が選択されてセレクタ６０ｌに出力される。従って、掃き出しエントリ３０６のデータが下位側に置かれ、連結エントリのデータが上位側に置かれて、データ３０２－０～１となる。

【０１０１】図２２は、バイトセレクタ６３ｂの構成を示す図である。６０ｒ、６０ｑは、リードポインタ３０３－０～１を基に、後段バイトマスクバッファの内容１１６－０～７から掃き出すエントリのバイトマスク３０４－０～１を取り出すセレクタである。６０ｎ、６０ｐは、バイトマスク３０４－０～１の値を基に有効なバイト位置のデータを取り出すセレクタである。

【０１０２】例えば、掃き出しエントリ３０６のデータ５８ｂが上位側にあるときは、リードポインタ３０３－１によってセレクタ６０ｒが働き、掃き出しエントリ３０６に対応するバイトマスクを選択し、セレクタ６０ｐに出力する。セレクタ６０ｐは、掃き出しエントリ３０６のデータ５８ｂの内、バイトマスク３０４－１で指示された有効なバイト位置のデータを取り出し、ＳＣＭストアデータ１０５として出力する。

【０１０３】〈実施例４〉次に、実施例４について説明する。キャッシュミスにより処理が止まることを回避する目的で、キャッシュメモリからではなく直接主記憶からデータを読み込み、ロードデータの到着を待たずに次の処理に移ることができるロード命令を導入する。本実施例では、このロード命令をｐｒｅｌｏａｄ命令と呼ぶことにする。ｐｒｅｌｏａｄ命令は大量のデータを扱う場合に有利である。

【０１０４】しかし、同じアドレスに対するストア命令とｐｒｅｌｏａｄ命令があった場合、前のストア命令をｐｒｅｌｏａｄ命令が追い越してしまうと、ｐｒｅｌｏａｄ命令は間違ったデータを読み込んでしまう。そこで、主記憶を参照する命令間の順序性を保障する必要がある。

【０１０５】図２３は、本発明の実施例４の構成を示す図である。１７は本実施例によって設けられたロードストアバッファである。ロードストアバッファ１７は、ｐｒｅｌｏａｄ命令とストア命令を保持するＦＩＦＯ型のバッファである。

【０１０６】図２４は、ロードストアバッファの構成を示す図である。６６は命令フィールドＯＰである。命令フィールド６６は、当該エントリの命令がｐｒｅｌｏａｄ命令かストア命令かを示す。

【０１０７】ロードストアバッファ１７は、本実施例では、８段のエントリからなり、各エントリは３２ビットのアドレスフィールド、６４ビットのデータフィールド、命令フィールドＯＰからなる。ストアアドレスとデータは信号線１０８から入力され、命令は信号線１２７から入力され、ライトエントリ１２２の示すエントリに書き込む。このとき、ｐｒｅｌｏａｄ命令ならば、データフィールドの値は意味を持たない。一方、ロードストアバッファ１７からの掃き出しは、リードエントリ１２３の示すエントリの、アドレス、データ、命令をセレクタ６０ｅで選択する。

【０１０８】選択した命令がｐｒｅｌｏａｄ命令ならば、主記憶に対して該命令が出力されるだけで、パイプライン遅延論理３には出力されない。ストア命令ならば、前述した実施例と同様に、主記憶に書き込むと同時に、パイプライン遅延論理３に送りＳＣＭヒット判定を行う。

【０１０９】

【発明の効果】以上、説明したように、本発明によれば、ストアバッファの入力と出力のピッチを一致させることができるので、ストアバッファに空きがなくなる状態の発生を低減することができ、その結果、ＣＰＵの性能低下を防ぐ効果がある。また、キャッシュヒットする

ストア命令のみストアバッファに入力するので、ストアバッファを効率よく使用できる効果がある。さらに、複数のストア命令によるデータ書き込みを同時に処理できるので、スループットを向上する効果がある。また、ストアスルー方式のキャッシュメモリを採用した場合、ストアバッファへの入力と主記憶への書き込みピッチが一致するので、主記憶に繋がるバスを効率よく使用できる効果がある。

【図面の簡単な説明】

【図１】本発明の実施例１の構成を示す図である。

【図２】実施例１の前段ストアバッファの構成を示す図である。

【図３】ＳＣＭヒット判定論理の動作例を示す図である。

【図４】実施例１のセット信号生成論理の構成を示す図である。

【図５】リードライトポインタの構成を示す図である。

【図６】実施例１の後段ストアバッファの構成を示す図である。

【図７】実施例１の動作例を示す図である。

【図８】本発明の実施例２の構成を示す図である。

【図９】実施例２の前段ストアバッファの構成を示す図である。

【図１０】実施例２の前段バイトマスクバッファの構成を示す図である。

【図１１】バイトマスク更新論理の構成を示す図である。

【図１２】後段バイトマスクバッファの構成を示す図である。

【図１３】実施例２の後段ストアバッファの構成を示す図である。

【図１４】バイトセレクタの構成を示す図である。

【図１５】アドレス一致論理の構成を示す図である。

【図１６】実施例２のセット信号生成論理の構成を示す図である。

【図１７】本発明の実施例３の構成を示す図である。

【図１８】連結エントリバッファの構成を示す図である。

【図１９】実施例３のアドレス一致論理の構成を示す図である。

【図２０】実施例３の後段ストアバッファの構成を示す図である。

【図２１】データセレクタの構成を示す図である。

【図２２】バイトセレクタの構成を示す図である。

【図２３】本発明の実施例４の構成を示す図である。

【図２４】ロードストアバッファの構成を示す図である。

【図２５】本発明の実施対象となる情報処理装置の構成図である。

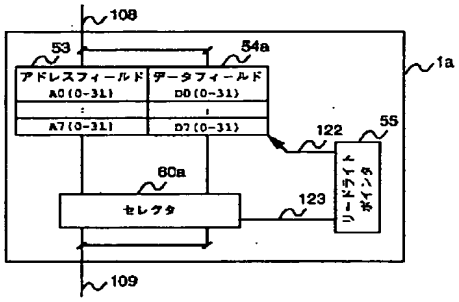
【図２６】ダイレクトマッピング法を説明する図である。

【符号の説明】

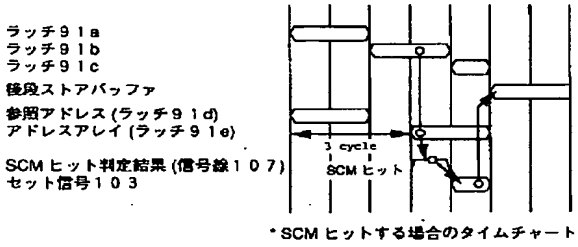
１ａ～ｂ　前段ストアバッファ
２ａ～ｃ　後段ストアバッファ
３　ＳＣＭヒット判定論理
４　主記憶
５ａ、ｂ　セット信号生成論理
６　データアレイ
７　アドレスアレイ
８　パイプライン遅延論理
９　アドレス比較器
１１　前段バイトマスクバッファ
１２　後段バイトマスクバッファ
１３　バイトマスク遅延論理
１４　バイトマスク更新論理
１５ａ、ｂ　アドレス一致検出論理
１６　連結エントリバッファ
１７　ロードストアバッファ
５１　後段ストアバッファライトポインタ
５２　更新論理
５３　アドレスフィールド
５４ａ、ｂ　データフィールド
５５　リードライトポインタ
５６ａ　ライトポインタ用ラッチ
５６ｂ　リードポインタ用ラッチ
５７ａ、ｂ　後段ストアバッファアドレスフィールド
５８ａ、ｂ　後段ストアバッファデータフィールド
５９　有効ビット
６０ａ～ｒ　セレクタ
６１　後段ストアバッファリードポインタ
６２　後段ストアバッファリードポインタ用ラッチ
６３ａ、ｂ　バイトセレクタ
６４、６５　アドレス比較器
６６　命令フィールド
６７　リードポインタ入れ換え論理
６８　データセレクタ
６９　デコーダ
７１　命令実行部
７２　１次キャッシュメモリ（ＦＣＭ）
７３　ストアバッファ
７４　ＣＰＵ
７５　２次キャッシュメモリ（ＳＣＭ）
９１ａ～ｆ　ラッチ
１０１　ＳＣＭアドレスアレイ参照アドレス
１０２　ＳＣＭアドレスアレイの内容
１０３　セット信号
１０４　ＳＣＭストアアドレス
１０５　ＳＣＭストアデータ
１０６　主記憶書き込みアドレスとデータ
１０７　ＳＣＭヒット判定結果

１０８　ストアアドレスとデータ
１０９　前段ストアバッファから出力したストアアドレスとデータ
１１０　遅延したストアアドレスとデータ
１１１　掃き出し要求
１１２　前段バイトマスク入力
１１３　前段バイトマスク出力
１１４　バイトマスク遅延論理出力
１１５－０～７　後段バイトマスク入力
１１６－０～７　後段バイトマスク出力
１１７　アドレス一致入力
１１８－０～７　アドレス一致出力
１１９－０～７　アドレスフィールドの内容
１２１　データ併合の有無
１２２　ライトエントリ
１２３　リードエントリ
１２４－０～７　有効ビット
１２５－０～７　連結エントリ
１２６－０～７　連結エントリバッファ出力
１２７　命令
１２８　ライト要求
１２９　リード要求
３０１、３０２　セレクトしたデータ
３０３　セレクト信号
３０５　データフィールド
３０６　掃き出しエントリ
３０７　データフィールドセット信号
３０８　選択したバイトマスク

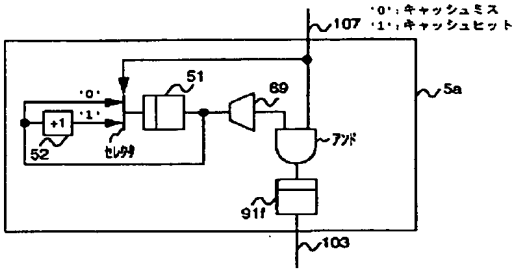
【図２】



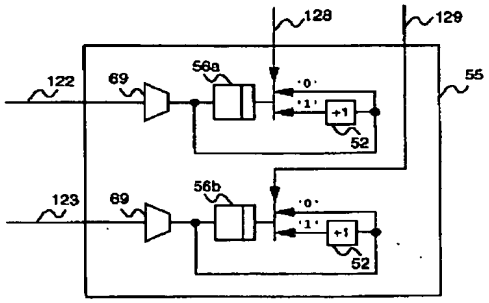
【図３】



【図４】

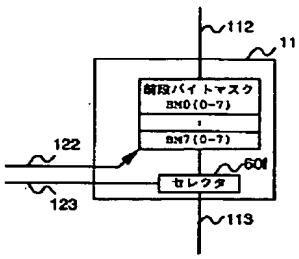


【図５】

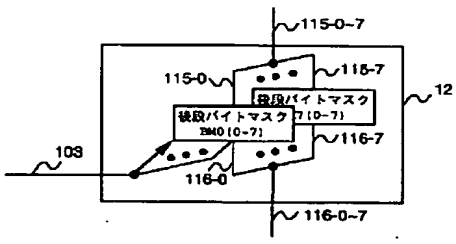


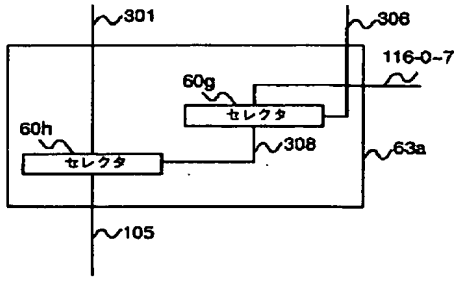
【図１４】

【図１０】

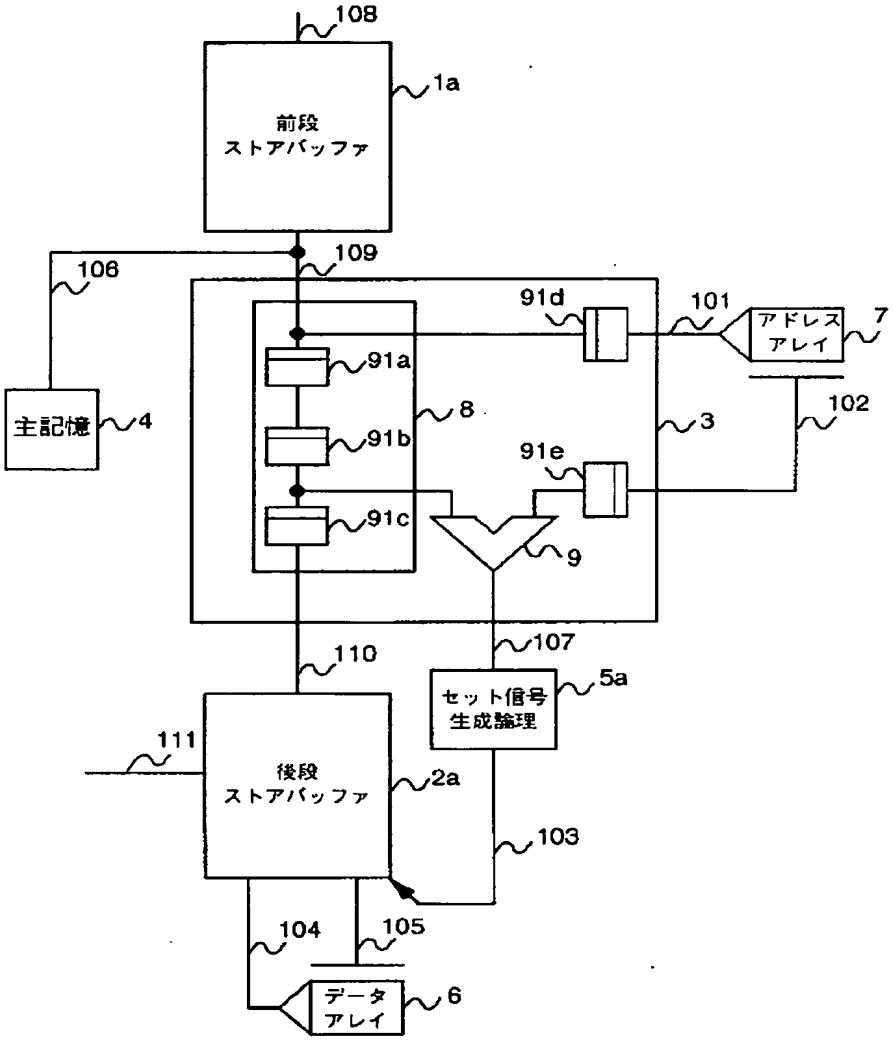


【図１２】

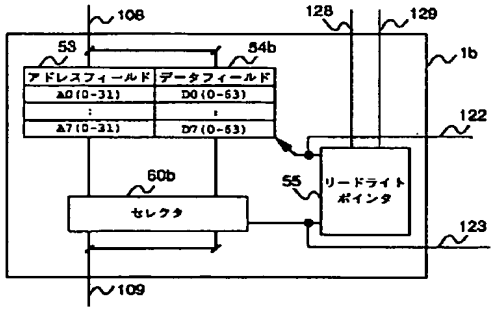




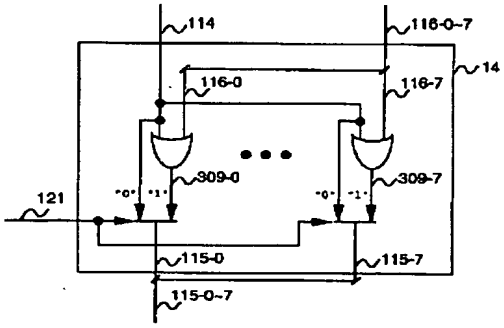
【図1】


108
1a
前段
ストアバッファ
109
106
91d
101
アドレス
アレイ
7
91a
8
3
102
主記憶
4
91b
91e
91c
9
110
107
セット信号
生成論理
5a
111
後段
ストアバッファ
2a
103
104
105
データ
アレイ
6

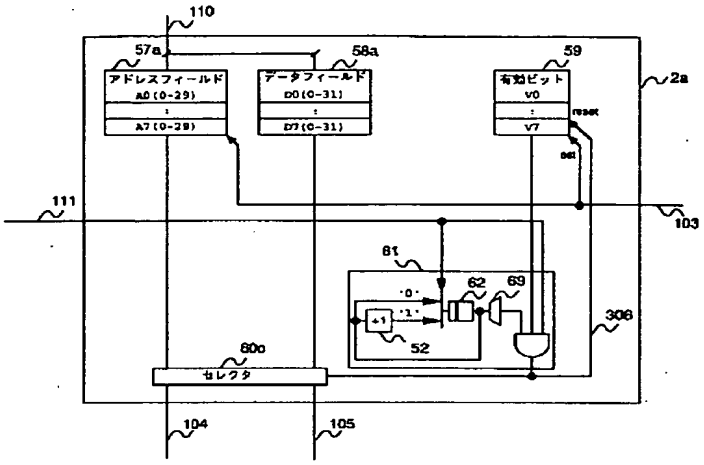
【図9】


108
128
129
53
54b
1b
アドレスフィールド
データフィールド
A0(0-31)
D0(0-63)
A7(0-31)
D7(0-63)
122
60b
55
リードライト
ポインタ
セレクタ
123
109

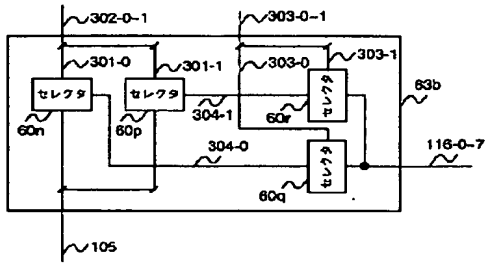
【図11】


114
116-0~7
116-0
116-7
14
309-0
309-7
"0"
"1"
121
115-0
115-7
115-0~7

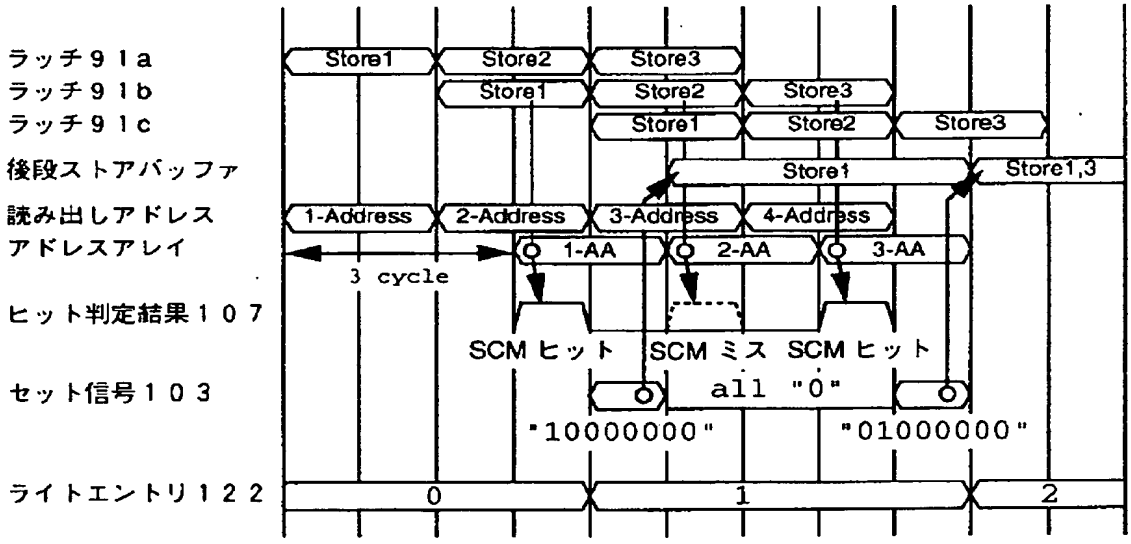
【図6】



【図22】

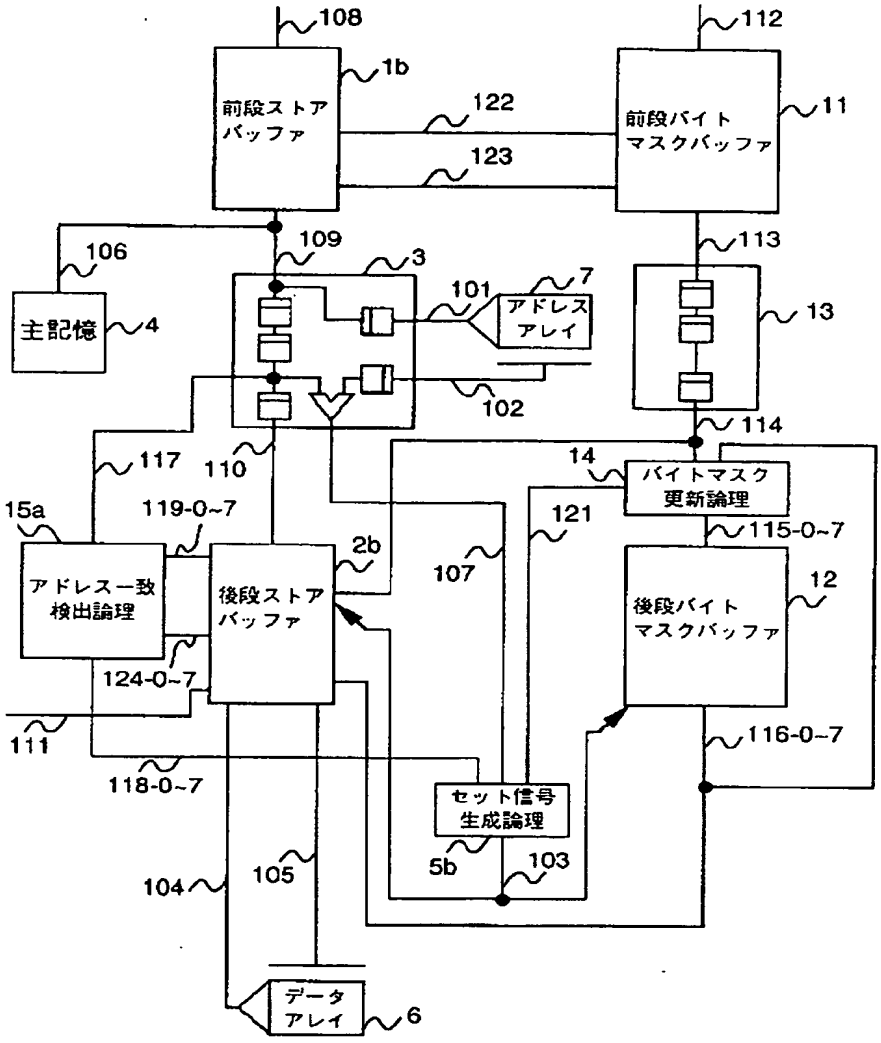


【図7】

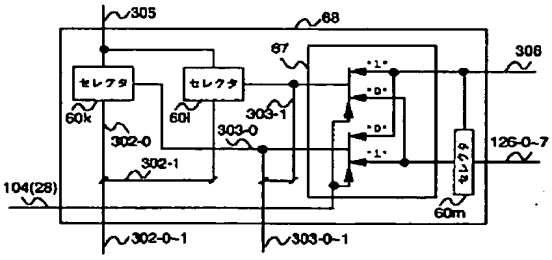


Store n　：n 番目のストア命令のストアアドレスとデータ
n-Address：n 番目のストア命令のアドレス
n-AA　　：n 番目のストア命令のアドレスアレイ

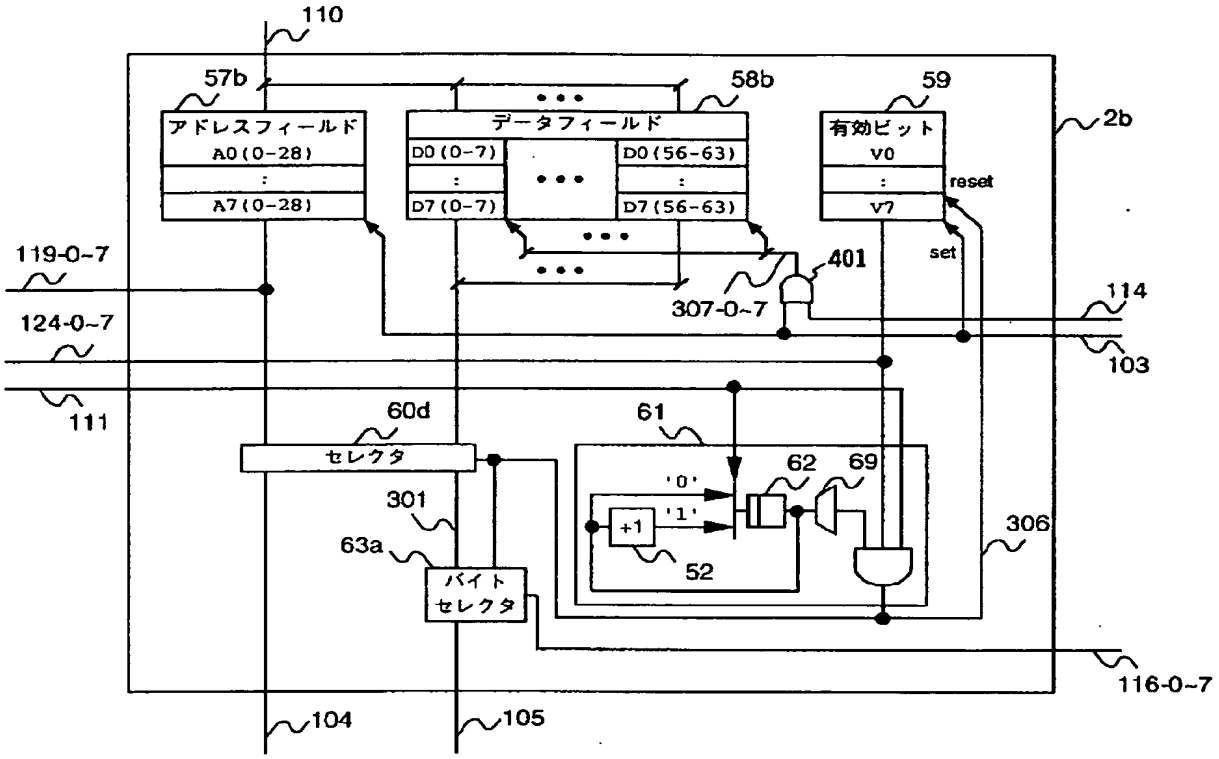
【図8】


108
1b
前段ストア
バッファ
122
123
112
前段バイト
マスクバッファ
11
106
109
3
113
主記憶
4
101
7
アドレス
アレイ
13
102
114
117
110
14
バイトマスク
更新論理
15a
119-0~7
121
115-0~7
2b
107
アドレス一致
検出論理
後段ストア
バッファ
後段バイト
マスクバッファ
12
124-0~7
111
116-0~7
118-0~7
セット信号
生成論理
5b
103
104
105
データ
アレイ
6

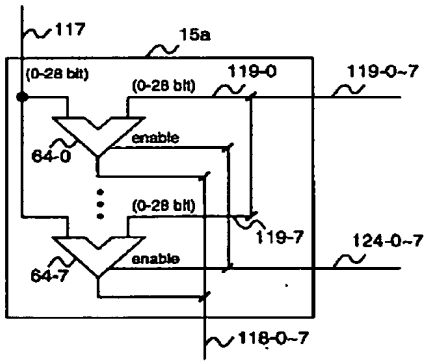
【図21】


305
68
67
306
セレクタ
セレクタ
60k
60l
303-1
303-0
302-0
302-1
126-0~7
104(28)
60m
302-0~1
303-0~1

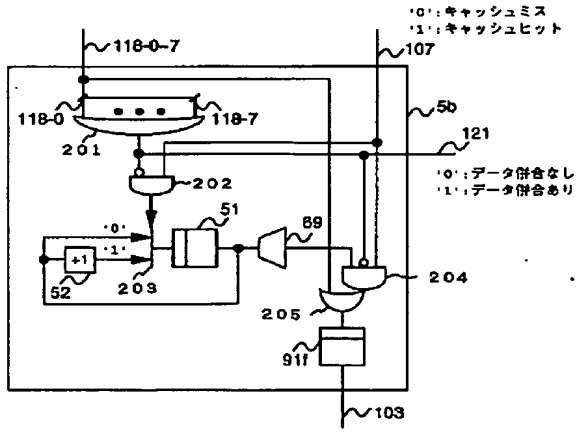
【図13】


110
57b
58b
59
2b
アドレスフィールド
A0(0-28)
A7(0-28)
データフィールド
D0(0-7)
D7(0-7)
D0(56-63)
D7(56-63)
有効ビット
V0
V7
reset
set
401
307-0~7
119-0~7
124-0~7
111
114
103
60d
セレクタ
61
62
69
'0'
'1'
+1
52
301
63a
バイトセレクタ
306
116-0~7
104
105

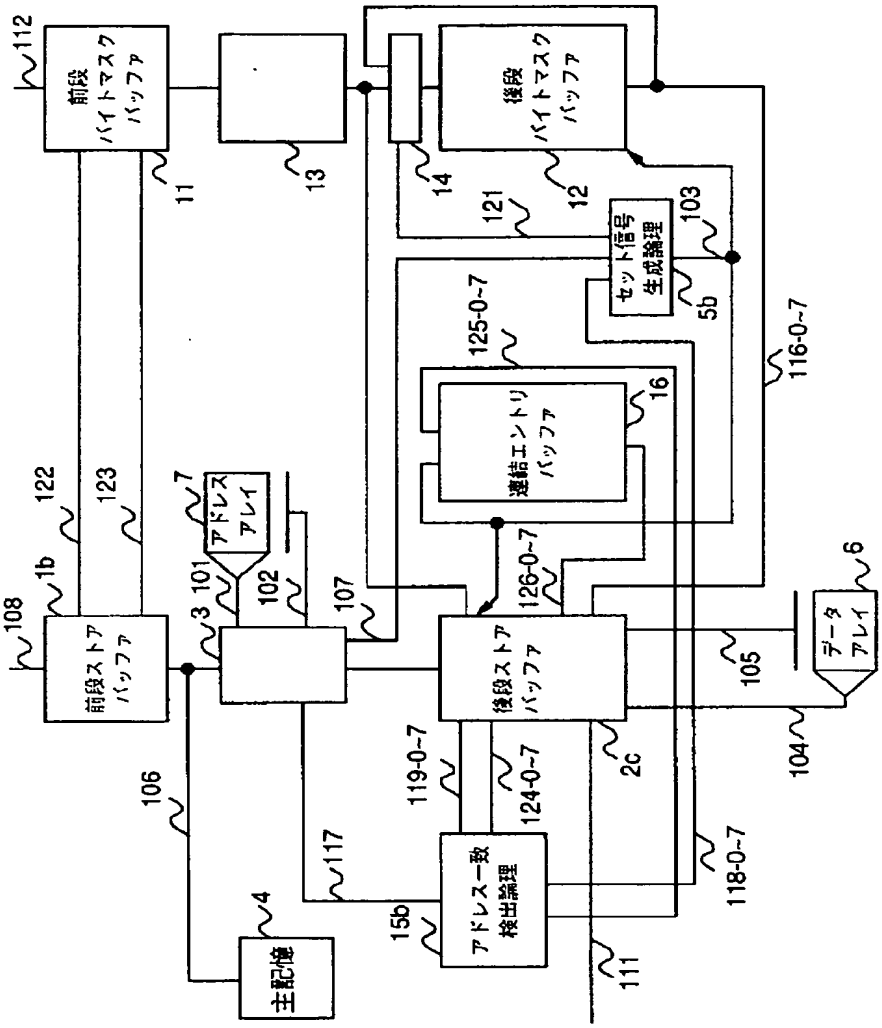
【図15】


117
15a
(0-28 bit)
(0-28 bit)
119-0
119-0~7
64-0
enable
(0-28 bit)
119-7
124-0~7
64-7
enable
118-0~7

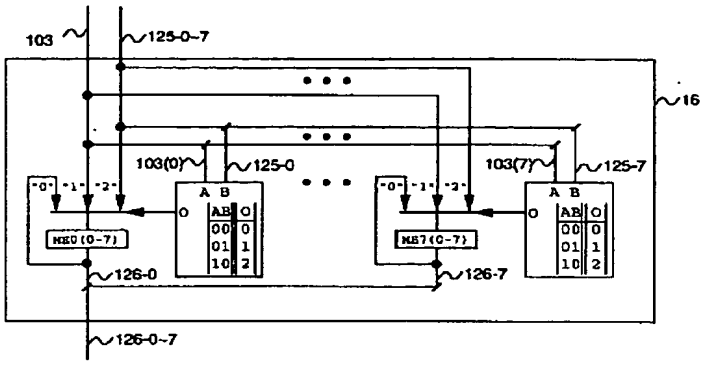
【図16】


'0':キャッシュミス
'1':キャッシュヒット
118-0~7
107
118-0
118-7
5b
121
201
202
'0':データ併合なし
'1':データ併合あり
51
69
'0'
'1'
+1
52
203
204
205
91f
103

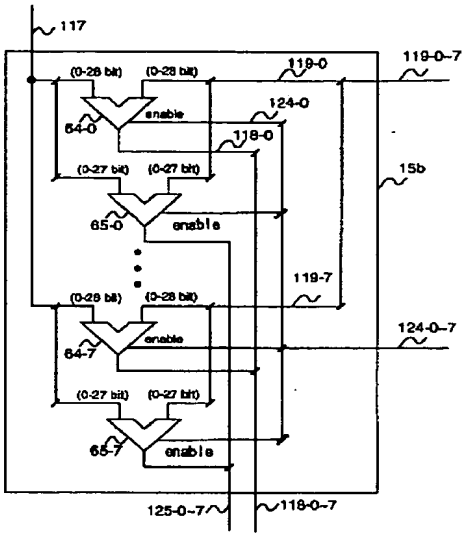
【図17】


112
前段
バイトマスク
バッファ
11
13
14
後段
バイトマスク
バッファ
12
121
セット信号
生成論理
103
5b
125-0~7
116-0~7
連結エントリ
バッファ
16
122
123
7
アドレス
アレイ
1b
101
102
107
126-0~7
6
108
3
前段ストア
バッファ
後段ストア
バッファ
データ
アレイ
105
104
106
119-0~7
124-0~7
2c
117
118-0~7
15b
アドレス一致
検出論理
4
主記憶
111

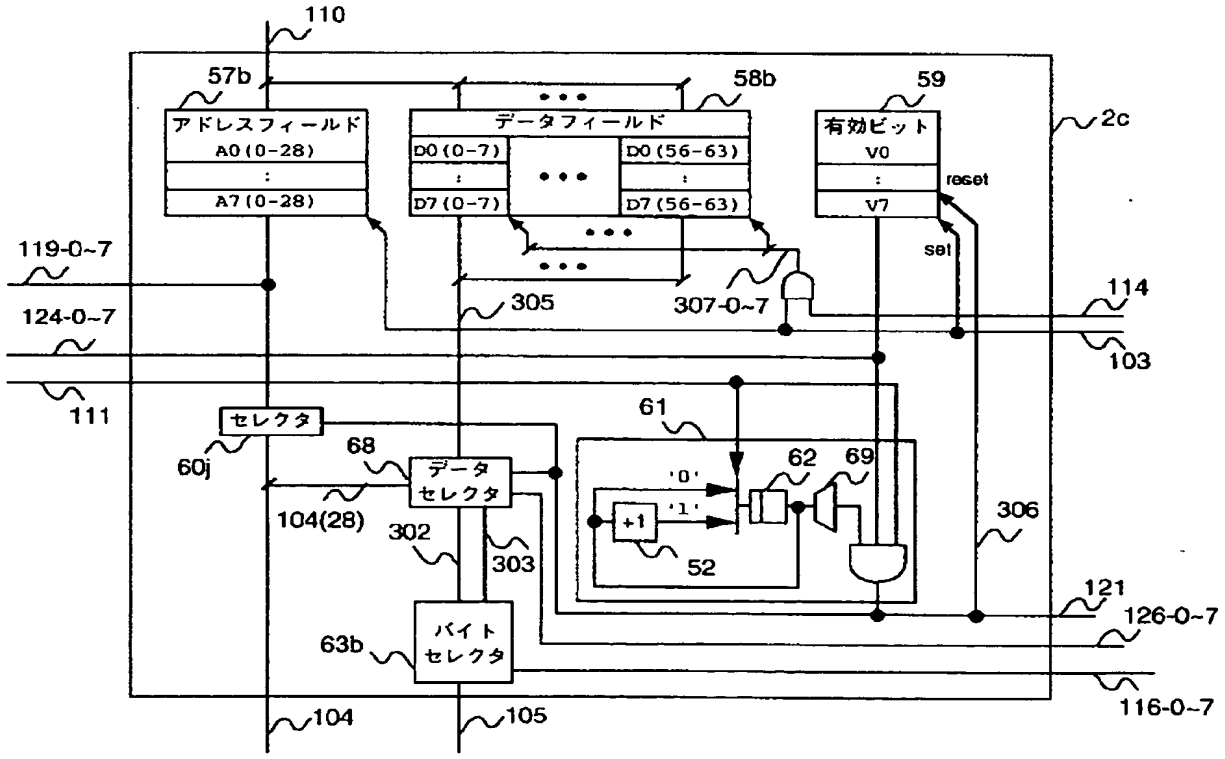
【図18】


103
125-0~7
16
103(0)
125-0
103(7)
125-7
A B
AB O
00 0
01 1
10 2
126-0
126-7
126-0~7

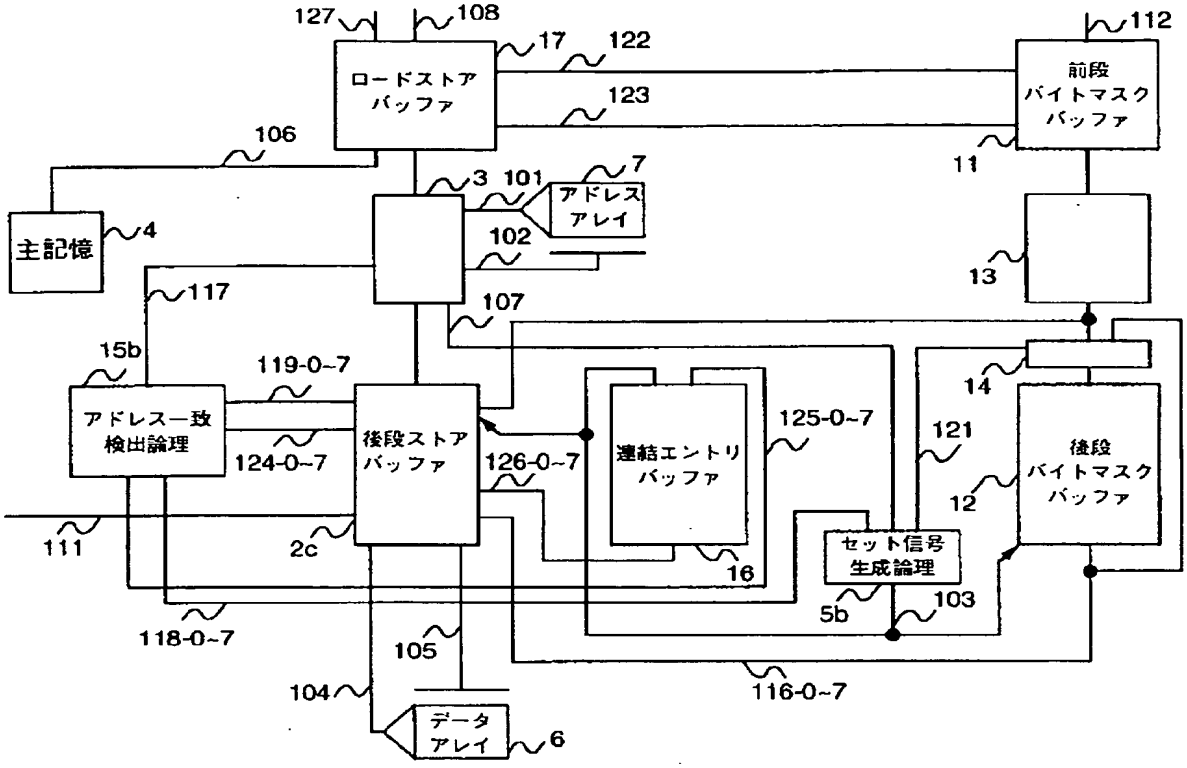
【図19】


117
119-0
119-0~7
(0-28 bit)
(0-27 bit)
enable
64-0
65-0
124-0
118-0
15b
119-7
64-7
65-7
124-0~7
125-0~7
118-0~7

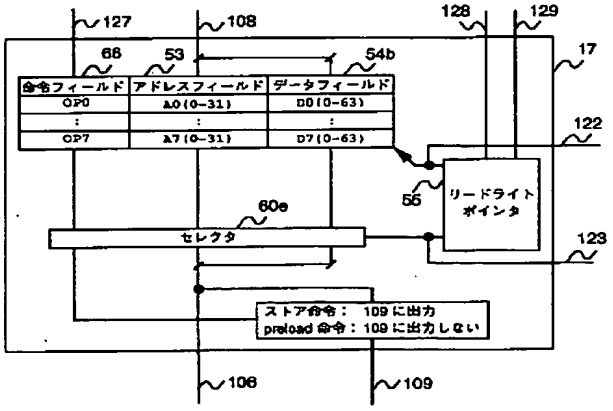
【図20】


110
57b
アドレスフィールド
A0(0-28)
A7(0-28)
58b
データフィールド
D0(0-7)
D7(0-7)
D0(56-63)
D7(56-63)
59
有効ビット
V0
V7
reset
sel
2c
119-0~7
124-0~7
111
305
307-0~7
114
103
セレクタ
60j
68
データ
セレクタ
104(28)
302
303
61
62
69
'0'
'1'
+1
52
306
121
126-0~7
63b
バイト
セレクタ
116-0~7
104
105

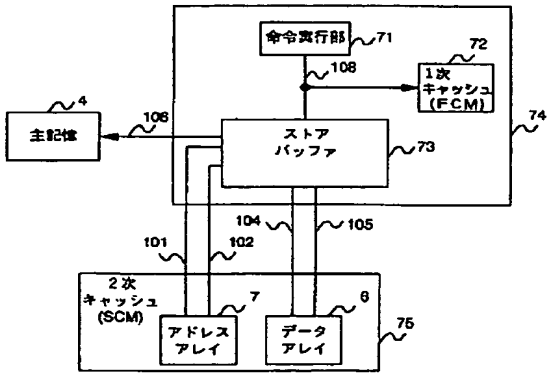
【図２３】


127
108
17
122
123
ロードストア
バッファ
112
前段
バイトマスク
バッファ
106
11
3
101
7
アドレス
アレイ
主記憶
4
102
13
117
107
15b
119-0~7
14
アドレス一致
検出論理
後段ストア
バッファ
125-0~7
121
連結エントリ
バッファ
後段
バイトマスク
バッファ
124-0~7
126-0~7
12
111
2c
16
セット信号
生成論理
5b
103
118-0~7
105
104
116-0~7
データ
アレイ
6

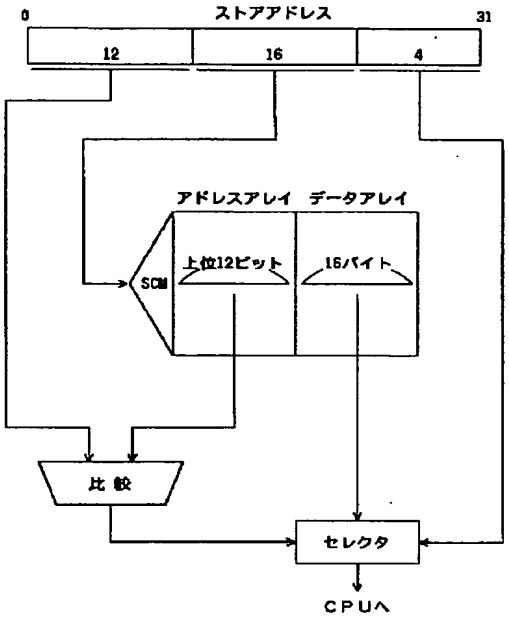
【図２４】


127
108
128
129
66
53
54b
17
命令フィールド
アドレスフィールド
データフィールド
OP0
A0(0-31)
D0(0-63)
OP7
A7(0-31)
D7(0-63)
122
55
リードライト
ポインタ
60e
セレクタ
123
ストア命令： 109に出力
preload命令： 109に出力しない
106
109

【図２５】


命令実行部
71
72
108
1次
キャッシュ
(FCM)
4
主記憶
106
ストア
バッファ
73
74
104
105
101
102
2次
キャッシュ
(SCM)
7
6
アドレス
アレイ
データ
アレイ
75

【図26】



フロントページの続き

(72)発明者　釜田　栄樹
神奈川県秦野市堀山下１番地　株式会社日立製作所汎用コンピュータ事業部内
(72)発明者　磯部　敏子
神奈川県秦野市堀山下１番地　日立コンピュータエンジニアリング株式会社内
(72)発明者　山本　敬
神奈川県秦野市堀山下１番地　株式会社日立製作所汎用コンピュータ事業部内
(72)発明者　上原　克利
神奈川県秦野市堀山下１番地　株式会社日立製作所汎用コンピュータ事業部内